Symbol Recognition in Elevation Drawings: A Survey

Bruce Bobier

bbobier@uoguelph.ca
CIS*6650
Department of Computing and Information Science
University of Guelph, Guelph, ON, N1G 2W1

Abstract

Thousands of elevation drawings are contained in archives worldwide, although in their current format, their contained semantic information is only accessible through manual inspection and their associated brief annotations. The introduction of a method to semantically interpret elevation drawings offers several knowledge contributions, including the enablement of semantic-based archival searches, an alternative method of input for CAD systems, and a flexible symbol recognition process that may be extended to other domains. This article discusses a two phase framework for interpreting elevation drawings and surveys the numerous algorithms and techniques that can be used during each phase for representing and classifying the contained architectural elements.

1 Introduction

Existing architectural elevation drawings are predominately archived in paper- or raster-based formats, and the tasks of indexing, retrieving and analysing these drawings is, by an large, a cumbersome manual process. Representing the drawings in a concise and accessible format will enable users and researchers to more easily access the rich information about the elevation drawings, such as the properties and dimensions of contained architectural elements, as well as provide the ability to conduct new forms of analysis and more specific queries.

Over the last three decades, research on symbol recognition in digital documents has primarily focused on geographic maps [43], technical drawings [30,56], mathematical equations [8], and diagrams [9], much of which is analogous to the problem of identifying and classifying architectural elements in elevation drawings. A symbol refers to graphic shape that can be distributed through

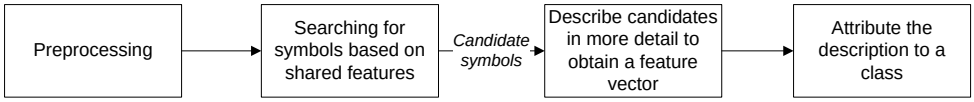

Fig. 1. Framework for interpreting elevation drawings.

a document and is characterized by its attributed semantics and rule-based construction [13]. From this, architectural elements such as doors, windows and stairs may be similarly considered as symbols in the realm of elevation drawings. Formally, an elevation drawing is a representation of a building's geometrical exterior as perceived from a horizontal viewpoint, without dimensional perspective. The drawings are manually created with pen and paper and digitized as 8-bit gray scale images using a scanner or overhead camera.

Currently, most content indexing and retrieval techniques rely on natural language processing approaches that operate on textual information, and there are several reasons to follow this paradigm for processing elevation drawings. Search engines are designed to process and interpret textual information using vocabulary control-based approaches, and a verbal interface that is created using thesauri of standard terms allows for the easier integration of content by search engines. Further, textual representations that are based on common terminology provide a familiar interface for users and reduce the amount of required learning. The second reason is due to the large precedence of alphanumeric information processing in computer-based environments, and following this direction may reduce the complexity and resource requirements for the development and usage of the system. This precedence includes a sizable library of algorithms and methodologies that provide a foundation for indexing and retrieving the elevation drawing contents and better enables a cohesive integration with existing systems. Finally, if the textual representation is sufficiently rich, it can be utilized by numerous CAD and information retrieval systems. This article discusses a two phase framework for interpreting elevation drawings and surveys the numerous algorithms and techniques that can be used during each phase for representing and classifying the contained architectural elements.

In general, the approach most commonly used for identifying and classifying symbols follows two sequential phases (Fig. 1). The representation phase is discussed in Section 2, and aims to transform the raster document into a form that is better suited to the phase's primary task of locating potential objects of interest. This phase also includes the interpretation of lines and arcs through vectorization or direct-recognition, which are discussed in Section 3. The classification phase seeks to classify the described symbols and is discussed in Section 4. Finally, Section 5 concludes and discusses areas for future research.

2 Representation Phase

The process of identifying and classifying symbols in a document begins with the representation phase, which aims to extract from the document, the information deemed to be most significant (e.g. black lines in an elevation drawing) for subsequent analytical tasks. Additionally, this phase attempts to reduce the amount of noise and data contained by the document, and to represent the underlying content as accurately as possible.

The remainder of this section discusses the approaches that are commonly used in the first part of this phase, namely preprocessing the document through binarization and text/graphics segmentation.

2.1 Preprocessing

Most digital elevation drawings are created from scanned versions of pen and paper drawings and frequently need to be preprocessed in order for the vectorization task to be successful. Presently, two common forms of preprocessing are discussed, namely binarization and text/graphic segmentation.

2.1.1 Binarization

Depending on the format and origin of the elevation drawing to be analysed, *binarization* (the process of converting a grey scale image to black and white), may or may not be required. In elevation drawings, the primary need for binarization is to remove unnecessary background information, such as paper texture, in order to reduce the complexity of the subsequent representation and classification. In many images, the black drawing ink bleeds in the paper, causing shades of grey to dilate around the intended line, while in others, the paper contained some wrinkles which are evident in the scanned version. Each of these occurrences result in an image having multiple grey values and thus require binarization.

In a recent comparative evaluation [6], 22 binarization algorithms were assessed for their performance (both in terms of quality and computation time) at binarizing a set of 60 images that characterize elevation drawings from the Historic American Building Survey [1]. In this study, clustering-based methods, which cluster the gray level samples into two groups as either foreground or background data, or alternatively by modeling the image as a mixture of two Gaussian distributions, were found to be most successful. Specifically, the clustering-based algorithms by Brink [7], Otsu [38], Ridler and Calvard [41], and Yanni and Horne [58], were found to have an accuracy exceeding 90%,

and to perform statistically significantly better than the other evaluated algorithms.

2.1.2 Text/Graphic Segmentation

Frequently elevation drawings contain textual annotations embedded in the drawing that describes attributes of the building, such as its construction materials, dimensions, name or function of the element, or notes from the artist. Although this textual layer can provide an additional source of semantic information, it also interferes with the process of recognizing the lines and building elements, as the problem of recognizing curvilinear text is computationally complex.

The process of segmenting text and graphics aims to separate the image into two layers that exclusively contain either the textual characters and annotations, or the graphical objects. Most authors perform this task early in the classification pipeline using image processing techniques, while the image is still in a raster format. A variety of methods have been presented to address this problem, and can be divided into three basic classes [52]:

- Locating linear shapes (assuming them to be graphics) using directional morphological filtering and removing all remaining curvilinear shapes (assuming them to be text),
- Locating and maintaining only lines based on the image's distance transform or vectorized representation, and
- Filtering the connected components through a set of rules to determine them to be graphics or text.

The latter class, introduced by Fletcher and Kasturi [20] and later improved by the LORIA group [52], has been the predominately selected approach for segmenting text and graphics in line drawings. Pragmatically, this class analyses the size and shape of the connected components in the image and groups the characters into strings using a Hough transform. The size and location of the strings' bounding boxes may be manually corrected and stored for subsequent character recognition. The remaining graphics layer is then further segmented using morphological filtering according to the detection of thin and thick lines, and the text is then iteratively eliminated from the resulting image of the graphics layer.

Methods in this class are well suited to the problem of segmenting text in technical and elevation drawings, as they scale well for processing many complex images, and are applied to raster-based images prior to vectorization, thereby reducing unnecessary subsequent computations [17,49,52]. However, they are less successful at separating text that contacts the graphical information (e.g. a string of characters placed atop a line), and therefore require

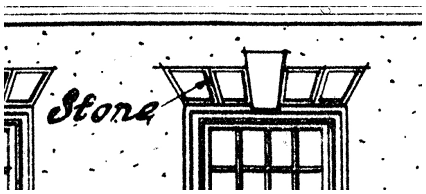

Fig. 2. Text that needs to be segmented from the graphics

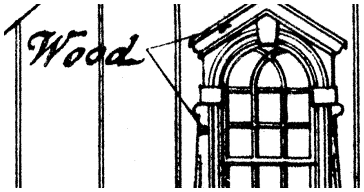

Fig. 3. Overlapping text/graphics that requires post-processing after segmentation

that post-processing be performed using a localized interpolation algorithm to rejoin any lines that were disconnected during segmentation. Fig. 2 provides an example of a textual annotation that indicates the construction material of the building element that will be easily removed prior to vectorization, and Fig. 3 gives an example that requires that post-processing be performed due to the overlapping text and graphics.

3 Line and Arc Interpretation

Once the document has been preprocessed through binarization and text/graphics segmentation, the next step in the representation phase, which aim to accurately represent the line and arc information, can follow. Existing representation methods can largely be classified into two paradigms: vectorization-based methods and direct-recognition methods. The former first converts the symbols into raw/low-level vectors and uses a vector-based recognition algorithm to represent them, while the latter attempts to conduct the recognition by operating directly on the raster image. This section discusses each of these paradigms and outlines their shortcomings.

3.1 Vectorization

Vectorization techniques have been proposed and developed for converting many types of line drawings, such as mechanical CAD drawings, schematic diagrams, line-and-box diagrams, and lines extracted using an edge detector [3,9,36,37]. These techniques are approximately divided into three classes: skeleton-based; matching opposite contours; and mesh-based. The following subsections outline each approach and discusses their strengths and weaknesses.

3.1.1 Skeleton-Based Methods

Skeleton-based approaches function by computing the medial axis, or topological skeleton, of a binary image by reducing the foreground image to a skeletal remnant that preserves the topology and connectivity of the original image, and discarding the majority of its foreground pixels. Within this class of methods, two well known paradigms can be defined, both of which require that the set of pixels constituting the drawing's medial axes be linked after skeletonisation, which is commonly performed using the algorithm described in [49].

The first skeletonisation paradigm iteratively thins the pixel data by converting some pixels from black to white, while maintaining the image's topological and morphological properties until only unit-wide lines exist [14]. The space complexity of these approaches is minimal, as only a few lines are buffered in memory at any time, making it scalable for processing a large number of complex drawings. However, as its iterative nature requires multiple passes of the image, the substantial time complexity reduces the method's ability to process large, complex image sets [49].

The second paradigm computes a skeleton of the image by preserving only those black pixels which constitute the centers of the maximal discs, while maintaining the connectivity of the original image [49]. The maximal discs are determined by calculating a distance transform for the pixel data in just two passes, resulting in a lesser time complexity than the previous skeletonisation paradigm. However, the space complexity of this approach limits its scalability, as it is difficult to compute a distance transform without maintaining the entire image in memory. This method has been applied in numerous studies for vectorizing line drawings (e.g. [2,3,9,49]) that frequently use the chamfer distance to approximate the Euclidean distance of the pixel data, following from Di Baja's [14] skeletonisation algorithm.

Each of these skeletonisation paradigms may be used to vectorize line drawings, although for the specific problem of architectural elevation drawings, the latter method of calculating the skeleton from the ridge of maximal disc centres is generally preferred. One factor influencing its popular usage is that the space complexity is only problematic when extremely large images are analysed, and the system's memory capacity is exceeded. To address this limitation, several methods have been presented to split up the image into square tiles, allowing the skeleton to be computed for each tile, after which the tiles can be merged together again [17,55]. However, both skeletonisation methods are very sensitive to noise, resulting in the false detection of "noisy" edges, and the frequent misplacement of the junction points of two lines. Because of the high level of geometric accuracy in elevation drawings, it is imperative that junction points be placed as accurately as possible, a requirement

which spurred the introduction of techniques that match opposite contours to address this problem.

3.1.2 Matching Opposite Contours

Vectorization methods based on matching opposite contours generally function in four steps [19,40,49]: detecting line contours in a drawing; polygonally approximating the lines; matching corresponding contours based on their slope and distance transforms; and computing the medial axis from the pairs of matched contours. The basic principle is to operate directly on the image's contours, rather than on the medial axis as skeleton-based methods do. During the labeling of connected components, the contours can be extracted and oriented to maintain the side of the contour on which the shape is located. Next, the contours are approximated with polygons, and the opposite segments of the same contour are matched together. The junction points of two lines are then calculated wherever two matched contours intersect an adjacent segment, thereby providing these approaches with more accurate junction placement than skeleton-based methods. Finally, the medial axis of the image is located by maintaining only the unit wide string of connected pixels located halfway between the matched opposite contours and using these lines to construct the vectorized image.

Compared with skeleton-based methods, contour-matching methods are more accurate at correctly assigning junction points and are less sensitive to noise [50]. However, in analysing complex drawings, contour-matching methods often have difficulty in resolving cases where several hypotheses may represent a given contour (e.g. in approximating arcs with multiple polygons), and require additional guidance by heuristics [29]. This class of methods work well on vectorizing straight lines, as well as thick lines, both of which are common in elevation drawings. However, contour matching methods often fail when attempting to match complex structures and require an extensive amount of computation without necessarily generating accurate results [49].

3.1.3 Mesh-Based Methods

The mesh-based or subsampling method was first introduced by Lin et. al [28], with numerous improvements made in [11,15,54]. These methods work by dividing the image into a number of $n \times n$ pixel meshes, with a value for n chosen such that each mesh should intersect with no more than one line. The difficulty in choosing a value for n is that it must be selected a priori, which requires it to be assigned either manually via a parameter, or through iterative attempts to automatically assign its value, both of which restrict the ability for automation with a cost of increased computation time.

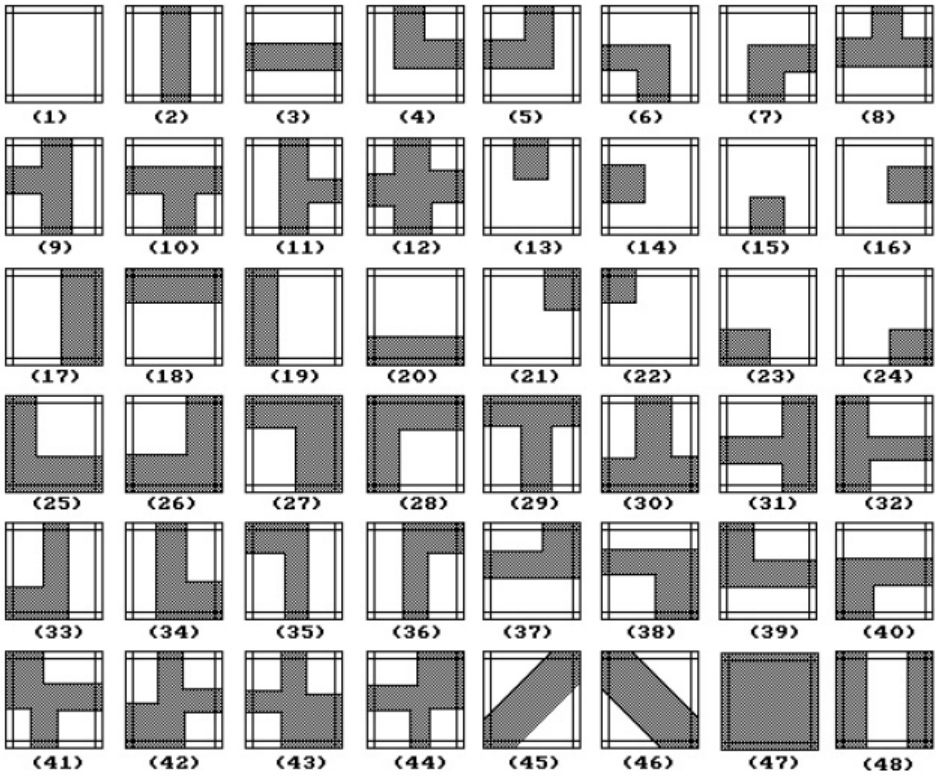

Fig. 4. 48 characteristic meshes used by mesh-based methods (reproduced from [54]).

The basic idea of mesh-based methods is to examine only the intersections of the sides of each mesh with other lines in the binary image, and using these observations to hypothesize the local line configurations [54]. The hypotheses are made by comparing each mesh with a set of 48 characteristic meshes (Figure 4, reproduced from [54]), and using a set of structural rules to extract a line vector based on the matched characteristic mesh. If a given mesh is unmatched with a characteristic mesh, it is recursively decomposed by extracting simple lines of black pixels from the mesh and attempting to match the newly formed mesh of extracted pixels until the original mesh is represented by a set of characteristic meshes. The medial axis of the extracted lines is then found by tracing along the known direction of the lines. An example of an image matched with a characteristic mesh is shown in Figure 5, where the perpendicular junction of two lines in the bottom left mesh is matched with characteristic mesh (25).

The granularity of these methods is based on the assigned mesh size, which typically results in a coarse vectorization of the image. In order to locate the lines and junctions with sufficient precision, several automated post-processing corrections are made to the coarse vectors to remove small barbs from the lines, merge together small line segments, and to merge several meshes (e.g. merging characteristic meshes (17), (18), (19), and (20) to create an area of solid colouring).

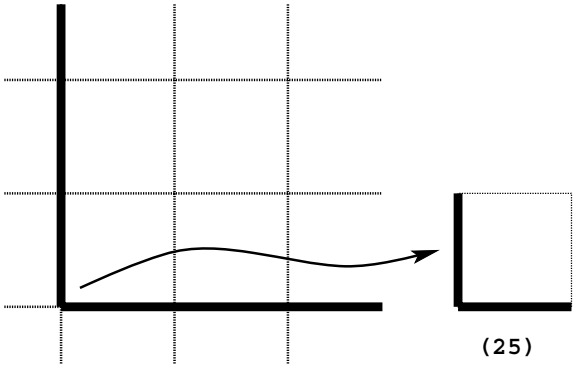

Fig. 5. Example image matched with a characteristic mesh

As with the other classes of vectorization methods, there are several strengths and shortcomings of the mesh-based methods. These methods work well at correctly positioning line edges with minimal displacement and are computationally efficient, as not all of the pixels are considered, merely those at the edges of the mesh. However, this efficiency also causes mesh-base methods to omit many small details (e.g. lines not accurately represented by a characteristic mesh) and to doubly detect a single edge in some cases (e.g. a thick line spanning multiple meshes). Further, in cases where a given mesh is not matched with one of the characteristic meshes, the accuracy of the resulting vector is reduced, as decomposing a complex mesh into a set of characteristic meshes struggles when irregular and oblique lines are encountered. The requirement of selecting an optimal mesh size is another shortcoming, as using an n value which is too small results in excessive computation, double detections of a single edge and overly complex lines, whereas selecting too large of an n value causes many meshes to be unclassified and small details to be omitted. Although these methods are suitable for vectorizing elevation drawings of simple linear buildings, they are not well suited for processing drawings with more detailed areas such as column capitals, fan lights and arched windows, or oblique lines, such as roofs and stone foundations.

3.2 Direct-Recognition Methods

Unlike vectorization-based methods, direct-recognition methods operate directly on the pixel level to recognize symbols from the raster image. These techniques are divided into three classes: pixel tracking; Hough transform; and region detection.

3.2.1 Pixel Tracking Methods

Pixel tracking methods were first introduced in [26] and are capable of recognizing both straight lines and arcs. Recognizing a straight line begins by locating two neighbouring foreground pixels in the image, and using this location and a few of its surrounding pixels to construct the narrowest strip of foreground pixels. Tracking then extends within this strip until no further four-connected pixels can be found within the strip. To recognize an arc, at least two connected straight lines must first be found in order to suggest the presence of an arc. Next, lines connecting each pair of inner and outer pixels are generated, and the intersections of these lines' bisectors are used to construct a centre polygon. Using this, if the original two straight lines are found to conform to the centre polygon, the arc may be extended [47].

The advantages of these methods are that both lines and arcs can be detected across line intersections, and that they are less sensitive to noise than other direct-recognition and vectorization-based methods. However, their success rests largely on the correct prediction of the initial direction of the strip, which cannot be ensured [47]. Additionally, these methods do not maintain line thickness.

3.2.2 Hough Transform Methods

Similar to pixel tracking, Hough transform-based methods are capable of directly recognizing both lines and arcs in raster images. Assessment of the entire image is computationally expensive, both in terms of time and spatial complexity, and to reduce this complexity, methods in this class select a subset of pixels (e.g. edge or connected points), called feature points, on which the transform is performed. The central advantage of these methods is their ability to easily and accurately recognize shapes in a noisy document. However, even though not all pixels are considered, calculating the Hough transform of just the feature points still requires a substantial amount computations to be performed, which often makes this class of methods too time consuming to be considered for symbol recognition in large documents. Again, as with pixel tracking methods, Hough transform-based methods have difficulty in recognizing line thickness [47].

3.2.3 Region-Based Methods

In [10], region-based methods were first proposed as an alternate means of directly recognizing straight lines in raster images using a maximal inscribing circle (MIC). This method follows the observation that, for a given line segment, the diameter of a MIC that inscribes the line, has a direction that is perpendicular to the line's orientation, provided that it has an absence of

noise in the continuous domain. The diameter of the MIC has two end points that are both edge pixels, which are indicative of the line's width, and thus the perpendicular of the diameter is hypothesized as being the orientation of the inscribed line. The diameter also provides the initial direction for the region test, wherein the testing region is comprised of all line segments originating from a pixel on the diameter and ending at the first encountered edge pixel. The approach then uses the region encountered at these edge pixels as a testing area, and iteratively tests these regions until the area currently being tested is not larger than the previous region. When the exit condition has been met, the final straight line is identified as the last region to have been tested [10,47].

The advantages of these methods are that distortions at junctions are minimal, and that line thickness is preserved. However, region-based methods fail to directly recognize arcs and dashed lines, and frequently misrecognize incorrect segments. Further, the iterative tests are computationally expensive, and the methods perform poorly in noisy environments and when handling complex intersections.

3.3 Quality of Representation

It has been long established that all vectorization-based methods have problems with handling junctions, and that none of the existing methods provide a perfect solution [49]. For instance, skeletonisation generates distortions, contour-matching often fails to pair matching contours in complex situations, while mesh-based methods struggle with oblique junctions and fine-grained details. The shortcomings of each of these methods can be attributed to their being driven by local data, where the vectors are yielded solely by the analysis of a local pixel area, which results in an interpretation that does not match that of the draftsperson. Further, since geometric constraints are not typically considered until the post-processing of the vectors, the distorted symbol representations created during vectorization are often of inconsistent orientation, which offsets or transforms the final output from its ground truth form [47].

Direct-recognition methods on the other hand, employ geometric knowledge directly during the pixel-level analysis and can better handle junctions and instances of touching or overlapping regions. Additionally, they are global data-driven methods, in that the final parameters are assigned with respect to the global feature, and thus can often provide less ambiguous representations. With the exception of the Hough transform-based methods, direct-recognition is generally more time efficient than vectorization, because as the the representation task progresses, pixels are erased once they have been represented, thereby simplifying the raster data. Although lines and arcs can be recog-

nized well, these methods cannot be applied to non-selfsimilar curves or arcs with very large or very small radii, both of which cases are better handled by vectorization-based methods. Additionally, direct-recognition methods leave short or thin lines unrecognized, which vectorization-based methods are capable of recognizing, provided that the short line segments are not perceived as noise.

3.4 Data Structures for Symbols

Once the information contained in a document has been described, either through vectorization or direct-recognition, the next challenge is to insert the structural description into a searchable data structure. Graphs in myriad forms have long been established as suitable data structures for representing structural descriptions [12], and are particularly common in representations of documents that primarily consist of straight lines.

Attributed relational graphs (ARGs) are among the most widely employed data structures for this task [13,39], where the nodes and edges in an ARG respectively describe the junctions and chain of connected points linking together two junction points [48]. This data structure represents a symbol as a set of geometric features and the spatial relations between them using relational attributes. Region adjacency graphs (RAGs), are an alternative data structure that are capable of capturing a significant amount of information, provided that the segmentation method used in the representation phase has divided the document into homogeneous regions [30]. If an accurate segmentation cannot be ensured (as is often the case in line drawings where a given line may belong to multiple regions), a RAG can be constructed where the attributes of the nodes describe the easily extracted features (e.g. lines, arc, etc.), and the edges describe the topological and geometric relations between the nodes [51].

4 Classification Phase

Following the representation phase, symbol classification may be performed, wherein unknown candidate symbols are classified as belonging to a predefined type or class of symbols. In this section, two classes of approaches are discussed, namely structural-based methods, which include template matching, graph matching, graph grammars, deformable template matching, and Hidden Markov Models, and statistical-based methods, including plain binary image, geometric features, and moment invariants, as well as methods to divide the feature space, including artificial neural networks, decision trees, and

similarity-based methods.

4.1 Structural-Based Classification

Template matching was among the earliest efforts in symbol recognition, and is a technique that is still applied for classifying candidate symbols when there is limited variability in the possible attributes (e.g. size, orientation, etc.) of a symbol. The primary limitation of template matching is that the accurate classification of symbols is largely scale and rotation dependent, such that the computational complexity may become prohibitive if many transformations must be computed and similarly evaluated. Additionally, the intrinsic nature of exact matching poorly handle instances where the symbols are touching or are occluded by lines or other symbols, as they are less flexible at recognizing distorted or inexact representations.

Graph matching techniques are more applicable to the classification of symbols in line drawings, or any document that contain noise or variable symbol shapes, orientations, scales, etc.. Methods in this class commonly use techniques such as isomorphism and graph-subgraph isomorphism for classifying candidate symbols that have been previously described with graphs during the representation phase. As the linear nature of line drawings is well suited for vector- or graph-based representations, graph matching techniques are an often employed technique for classification of symbols in these documents [12]. Inexact symbols are common to this type of document due to the document noise and variability of hand-drawn symbols, both of which may have been exacerbated during digitization and in the previous phases. To aid in the matching process, distance-based methods [28] are commonly employed to evaluate the distance, or alternatively, the similarity, between a candidate symbol and prototype. Error-correcting isomorphism (the minimum cost of transformations required to distort a candidate symbol to the prototype symbol), is one such distance that may be used to compute a distance value [33,34]. These transformations, or edit operations, include the insertion, deletion and substitution of both vertices and edges [31], with each operation having been previously assigned a cost. Using the set of operations and their associated costs, the algorithm operates by finding the sequence of transformations that minimize the cost of distorting the prototype graph to match the candidate graph. The Hausdorff distance is an alternative metric that has been shown to be robust in noisy documents and effective for computing the distance between a candidate symbol and prototype [27,39,46]. However, the computational complexity of the Hausdorff distance is substantial, as it requires a complete search for matching, and also has limitations in large geometrical transformations [39].

Although inexact graph matching techniques are generally better suited for

recognizing symbols in line drawings than exact template matching techniques, their larger time and spatial complexity is a significant limitation. However, by clearly defining the prototypes in an efficiently searchable data structure, the time complexity can be greatly reduced, although the spatial complexity of maintaining many prototypes may remain prohibitive.

Methods using formal grammars, specifically graph grammars, are another class of structural approaches that attempt to classify previously described vectorial or graph-based candidate symbols [25,32,44]. These methods use a grammar consisting of production rules, terminals, non-terminals, and a start symbol to represent all valid forms that a symbol may take. The classification phase for a candidate symbol consists of parsing the symbol's vectorial or graph-based model to determine if it can be generated by a prototypical grammar. To perform inexact matching, various types of error-correcting parsers have been introduced [22]. Grammar-based methods are best suited to classification tasks where the symbols generally follow a standardized notation and thus can be unambiguously specified by a grammar [32].

Another prominent type of structural approaches are those which use deformable template matching [23,53] in an attempt to classify a candidate symbol by finding a deformation of a symbol prototype that closely matches it. The classification is conducted by minimizing an energy function that is composed of an internal energy that measures the degree of deformation of the prototype, and an external energy that measures the degree of similarity between the deformed prototype and the candidate symbol. These approaches are known to be efficient at classifying symbols in hand-drawn images [4], although a shortcoming is that the scalability is not guaranteed when the number of prototypes increases.

Hidden Markov Models (HMMs) are another set of structural methods, wherein each symbol's structure is represented by a sequence of states which, when combined, generate the image. Generally, the problem of classifying a candidate symbol involves discovering the sequence of discrete states that have the highest probability of generating the symbol. HMMs have the advantage of being able to support numerous features to represent symbols, to segment the symbols, and to classify distorted symbols [30].

4.2 Statistical-Based Classification

Whereas structural approaches typically operate on vector-based representations of a symbol previously decomposed into vectorial primitives, with statistical approaches, the primitives are usually represented by an n-dimensional feature vector [32]. For these approaches, classification is performed by di-

viding the feature space into distinct classes, such that each symbol belongs to a unique class. The first step is to define the statistical descriptors with respect to the properties of the symbols to be classified. The primary goals of this step are to minimize the within-class distance of related symbols, and to maximize the distance between disjoint symbol classes. The remainder of this section briefly discusses the three major groups of statistical descriptors: plain binary image, geometric features, moment invariants, shape context, and image transformations, as well as three methods for partitioning the feature space: artificial neural networks, decision trees, and similarity-based methods.

The most basic statistical descriptor is the plain binary image, in which each pixel in the image constitutes a single feature in the feature vector [45]. The main advantages of these descriptors is the low complexity of the feature space, which also maps directly to the visual image. However, they are neither rotation- nor scale-invariant and are very sensitive to distortion and noise [32,57]. Methods using geometric features such as area, holes and intersections, have feature space dimensions that are much smaller than those of the plain binary image descriptor. A difficulty of using these descriptors is in making the correct selection of geometric features that are appropriate for the document type, as this is crucial for discrimination of symbols with affine transformation invariance. A third group of statistical descriptors that have been applied to symbol classification problems are moment invariants [21,35], which offer the advantages of being scale- and rotation-invariant and easily computed. However, this descriptors in this group have the disadvantage of being unable to profile the symbol's structure in detail [57].

Once a set of descriptors has has been selected, the second phase of classification involves the selection of a method to partition the features space and to classify the feature vectors to a predefined symbol class [32]. The task of partitioning the features space is usually performed using artificial neural networks (ANNs), decision trees, or similarity-based methods.

Artificial neural networks have been applied to classification tasks in numerous problem domains with good results [5]. With these methods, learning is performed automatically via a training set of feature vectors, for which the ANN attempts to learn the optimal parameters to the network for classification. This learning ability is a central advantage of ANNs, as it allows them to adapt to the properties of the training set, thereby enabling greater flexibility for future classification.

Decision trees, such as the C4.5 classifier, are also capable of performing symbol classification, where each tree node represents a specific condition about the value of a particular feature in the feature vector [32]. With decision trees, classification is performed by recursively traversing the tree in accordance to the results of each node's condition test on the input symbol. Once a leaf node

has been reached, the traversal halts and the candidate symbol is classified as an instance of that leaf node.

Similarity-based methods are the most basic method to partition the feature space and, similar to structural graph-matching techniques, involve defining a distance function over the feature vectors. The function is then used to classify each candidate symbol's feature vector as belonging to the symbol class having the nearest prototype. A related partitioning method involves using the k-nearest neighbours, wherein several prototypes of each class are selected, and for each candidate symbol, a set of the k-nearest prototypes is constructed. From these, classification is performed by assigning the candidate symbol to the class having the most representatives in the set of k-nearest prototypes.

5 Conclusion

This article has presented a survey of symbol recognition approaches for classification of architectural elements in elevation drawings by dividing the problem into two phases: representation and classification. Given that each surveyed vectorization-based and direct-recognition approach comes with its own strengths and limitations, one of the most promising approaches to the representation phase involves the construction of a stable and efficient combination of vectorization-based and direct-recognition methods that emphasizes the strengths of each and mitigates their weaknesses. For example, such an approach may first apply a direct-recognition technique such as pixel-tracking to represent straight lines, arcs and circles, and erasing them after recognition. Next, a vectorization-based technique such as skeletonisation could be applied to represent the remaining curves, short/thin lines and small arcs. Several advantages of this combined approach may be noted, including improved computational efficiency due to the erasure of recognized segments, as well as the more accurate placement of junction points from their direct-recognition.

Regardless of the vectorization method that is selected, a certain amount of post-processing is required to improve the accuracy of approximating arcs and correct X, L, T, and Y shaped junction points, and to simplify the number of nodes in the lines. These post-processing techniques have been consciously omitted from the scope of this article, as they alone constitute a significant literature review. Future research may conduct a comparison of methods to correct junction points (e.g. [24]), simplify lines (e.g. [18]), and approximate arcs (e.g. [15,16,42]). Finally, numerous data structures have been proposed for storing the symbol prototypes, such as ontologies, databases, hierarchies, etc., and it too merits the attention of a full comparative article in future work.

References

[1] Historic American Buildings Survey/Historic American Engineering Record (HABS/HAER). http://memory.loc.gov/ammem/collections/habs_haer, 2007.

[2] Ablameyko, S., Bereishik, V., Frantskevich, O., Homenko, M., Melnik, E., Okun, O., and Paramonova, N. A system for automatic vectorization and interpretation of map-drawings. In *IEEE International Conference on Computer Vision* (1993), pp. 456–460.

[3] Ablameyko, S., Bereishik, V., Paramonova, N., Marcelli, A., Ishikawa, S., and Kato, N. Vectorization and representation of large-size 2-D line drawing images. *Journal of Visual Communication and Image Representation 5* (1994), 245–254.

[4] Ah-Soon, C., and Tombre, K. Architectural symbol recognition using a network of constraints. *Pattern Recognition Letters 22*, 2 (2001), 231–248.

[5] Altuwaijri, M., and Bayoumi, M. A thinning algorithm for Arabic characters using Art2 Neural-Network. 260–264.

[6] Bobier, B. Evaluation of binarization algorithms for preprocessing elevation drawings. *Unpublished*, University of Guelph, Guelph, ON, March 2007.

[7] Brink, A. Gray-level thresholding of images using a correlation criterion. *Pattern Recognition Letters 9*, 5 (June 1989), 335–341.

[8] Chan, K.-F., and Yeung, D.-Y. Mathematical expression recognition: a survey. *International Journal on Document Analysis and Retrieval 3*, 1 (2000), 3–15.

[9] Chen, Y., Langrana, N. A., and Das, A. Perfecting vectorized mechanical drawings. *Computer Vision and Image Understanding 63*, 2 (March 1996), 273–286.

[10] Chiang, J. Y., Tue, S., and Leu, Y. A new algorithm for line image vectorization. *Pattern Recognition 31*, 10 (October 1998), 1541–1549.

[11] Collin, S., and Colnet., D. Syntactic analysis of technical drawing dimensions. *International Journal of Pattern Recognition and Artificial Intelligence 8*, 5 (1994), 1131–1148.

[12] Conte, D., Foggia, P., Sansone, C., and Vento, M. Thirty years of graph matching in pattern recognition. *International Journal of Pattern Recognition and Artificial Intelligence 18*, 3 (2004), 265–298.

[13] Cordella, L. P., and Vento, M. Symbol recognition in documents: a collection of techniques? *International Journal on Document Analysis and Recognition 3*, 2 (2000), 73–88.

[14] DiBaja, G. S. Well-shaped, stable and reversible skeletons from the (3,4)-distance transform. *Journal of Visual Communication and Image Representation 5*, 1 (March 1994), 107–115.

[15] Dori, D., Liang, Y., Dowell, J., and Chai, I. Sparse pixel recognition of primitives in engineering drawings. *Machine Vision and Applications 6*, 1 (1993), 69–82.

[16] Dosch, P., Masini, G., and Tombre, K. Improving arch detection in graphics recognition. In *Proceedings of the 15th International Conference on Pattern Recognition* (Barcelona, Spain, September 2000).

[17] Dosch, P., Tombre, K., Ah-Soon, C., and Masini, G. A complete system for analysis of architectural drawings. *International Journal on Document Analysis and Recognition 3*, 2 (Dec. 2000), 102–116.

[18] Douglas, D. H., and Peucker, T. K. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Canadian Cartographer 10*, 2 (1973), 112–122.

[19] Fan, K.-C., Chen, D.-F., and Wen, M.-G. Skeletonization of binary images with nonuniform width via block decomposition and contour vector matching. *Pattern Recognition 31* (July 1998), 823–838.

[20] Fletcher, A., and Kasturi, R. A robust algorithm for text string separation from mixed text/graphics images. *IEEE Transactions on Pattern Analysis and Machine Intelligence 10*, 6 (1998), 910–918.

[21] Hu, M. Visual pattern recognition by moment invariants. *Information Theory 8*, 2 (February 1962), 179–187.

[22] Ivanov, Y. A., and Bobick, A. F. Recognition of visual activities and interactions by stochastic parsing. *IEEE Transactions on Pattern Analysis and Machine Intelligence 22*, 8 (2000), 852–872.

[23] Jain, A. K., Zhong, Y., and Lakshmanan, S. Object matching using deformable templates. *IEEE Transactions on Pattern Analysis and Machine Intelligence 18*, 3 (1996), 267–278.

[24] Janssen, R. D., and Vosspoel, A. M. Adaptive vectorization of line drawing images. *Computer Vision and Image Understanding 65*, 1 (1997), 38–56.

[25] Kiyko, V. M. Recognition of objects in images of paper based line drawings. In *ICDAR '95: Proceedings of the Third International Conference on Document Analysis and Recognition (Volume 2)* (Washington, DC, USA, 1995), IEEE Computer Society, p. 970.

[26] Kovalevsky, V. New definition and fast recognition of digital straight segments and arcs. In *International Conference on Pattern Recognition* (1990), vol. 2, pp. 31–34.

[27] Kwon, O., Sim, D., and Park, R. Robust Hausdorff distance matching algorithms using pyramidal structures. *Pattern Recognition 34*, 10 (October 2001), 2005–2013.

[28] Lin, X., Shimotsuji, S., Minoh, M., and Sakai, T. Efficient diagram understanding with characteristic pattern detection. *Computer Vision, Graphics, and Image Processing 30*, 1 (April 1985), 84–106.

[29] Lladós, J., Martí, E., and López-Krahe, J. Architectural floor plan analysis. http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/LLADOS1/archit.html, 2004.

[30] Lladós, J., Martí, E., and Villanueva, J. J. Symbol recognition by error-tolerant subgraph matching between region adjacency graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence 23*, 10 (2001), 1137–1143.

[31] Lladós, J., and Sanchez, G. Symbol recognition using graphs. In *International Conference on Image Processing* (2003), vol. 2, pp. 49–52.

[32] Lladós, J., Valveny, E., Sánchez, G., and Martí, E. Symbol recognition: Current advances and perspectives. In *Graphics recognition, algorithms and systems* (2001), pp. 104–127.

[33] Lourens, T., and Würtz, R. P. Extraction and matching of symbolic contour graphs. *International Journal of Pattern Recognition and Artificial Intelligence 17*, 7 (2003), 1279–1302.

[34] Maple, C., and Wang, Y. A three-dimensional object similarity test using graph matching techniques. In *Proceedings of the Eighth International Conference on Information Visualisation* (Washington, DC, USA, 2004), IEEE Computer Society, pp. 363–369.

[35] Mindru, F., Tuytelaars, T., Gool, L. V., and Moons, T. Moment invariants for recognition under changing viewpoint and illumination. *Computer Vision and Image Understanding 94*, 1-3 (2004), 3–27.

[36] Noack, R. Converting CAD drawings to product models. Master's thesis, Royal Institute of Technology, 2001.

[37] Notowidigdo, M., and Miller, R. C. Off-line sketch interpretation. In *Proceedings of AAAI Fall Symposium on Making Pen-Based Interaction Intelligent and Natural* (October 2004), American Association for Artificial Intelligence.

[38] Otsu, N. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics 9*, 1 (1979), 62–66.

[39] Park, B. G., Lee, K. M., Lee, S. U., and Lee, J. H. Recognition of partially occluded objects using probabilistic ARG (attributed relational graph)-based matching. *Computer Vision and Image Understanding 90*, 3 (2003), 217–241.

[40] Ramel, J.-Y., Vincent, N., and Emptoz, H. A coarse vectorization as an initial representation for the understanding of line drawing images. In *GREC '97: Selected Papers from the Second International Workshop on Graphics Recognition, Algorithms and Systems* (London, UK, 1998), Springer-Verlag, pp. 48–57.

[41] Ridler, T., and Calvard, S. Picture thresholding using an iterative selection method. *System, Man and Cybernetics 8*, 8 (August 1978), 629–632.

[42] Rosin, P. L., and West, G. A. W. Segmentation of edges into lines and arcs. *Image Vision Computing 7*, 2 (1989), 251–270.

[43] Samet, H., and Soffer, A. MARCO: MAp retrieval by COntent. *IEEE Transactions on Pattern Analysis and Machine Intelligence 18*, 8 (1996), 783–798.

[44] Sánchez, G., and Lladós, J. A graph grammar to recognize textured symbols. In *ICDAR '01: Proceedings of the Sixth International Conference on Document Analysis and Recognition* (Washington, DC, USA, 2001), IEEE Computer Society, p. 465.

[45] Schürmann, J. *Pattern classification: a unified view of statistical and neural approaches*. John Wiley & Sons, Inc., New York, NY, USA, 1996.

[46] Sim, D., Kwon, O., and Park, R. Object matching algorithms using robust Hausdorff distance measures. *IEEE Transactions on Image Processing 8*, 3 (March 1999), 425–429.

[47] Song, J., Lyu, M. R., Cai, M., and Cai, S. Graphic object recognition from binary images: a survey and performance comparison. *Transactions on Systems, Man and Cybernetics: Applications and Reviews* (under review).

[48] Tabbone, S., and Wendling, L. Recognition of symbols in grey level line-drawings from an adaptation of the radon transform. In *ICPR '04: Proceedings of the 17th International Conference on Pattern Recognition* (Washington, DC, USA, 2004), vol. 2, IEEE Computer Society, pp. 570–573.

[49] Tombre, K., Ah-Soon, C., Dosch, P., Masini, G., and Tabbone, S. Stable and robust vectorization: How to make the right choices. *Lecture Notes in Computer Science 1941* (1999), 3–17.

[50] Tombre, K., and Tabbone, S. Vectorization in graphics recognition: To thin or not to thin. In *Proceedings of the 15th International Conference on Pattern Recognition, Barcelona (Spain)* (2000), vol. 2, pp. 91–96.

[51] Tombre, K., Tabbone, S., and Dosch, P. Musings on Symbol Recognition. In *Graphics Recognition—Ten Years Review and Future Pespectives*, W. Liu and J. Lladós, Eds., vol. 3926 of *Lecture Notes in Computer Science*. Springer Verlag, 2006, pp. 23–34.

[52] Tombre, K., Tabbone, S., Pélissier, L., Lamiroy, B., and Dosch, P. Text/graphics separation revisited. In *Proceedings of the Fifth International Association for Pattern Recognition* (August 2002).

[53] Valveny, E., and Martí, E. Application of deformable template matching to symbol recognition in hand-written architectural drawings. In *ICDAR '99: Proceedings of the Fifth International Conference on Document Analysis and Recognition* (Washington, DC, USA, 1999), IEEE Computer Society, p. 483.

[54] Vaxivière, P., and Tombre, K. Subsampling: A structural approach to technical document vectorisation. *Syntactic and Structural Pattern Recognition* (1995), 323–332.

[55] Vossepoel, A. M., Schutte, K., and Delanghel, C. F. Memory efficient skeletonization of utility maps. In *Proceedings of Fourth International Conference on Document Analysis and Recognition* (August 1997), pp. 797–800.

[56] Yan, L., and Wenyin, L. Engineering drawings recognition using a case-based approach. In *ICDAR '03: Proceedings of the Seventh International Conference on Document Analysis and Recognition* (Washington, DC, USA, 2003), IEEE Computer Society, p. 190.

[57] Yang, S. Symbol recognition via statistical integration of pixel-level constraint histograms: A new descriptor. *IEEE Transactions on Pattern Analysis and Machine Intelligence 27*, 2 (2005), 278–281.

[58] Yanni, M., and Horne, E. A new approach to dynamic thresholding. In *EUSIPCO European Signal Processing Conference* (1994), vol. 1, pp. 34–44.